Week 6

Review of Arrays

Manipulating an array of primitives

Manipulating an array of objects

Applets and Java Coordinate System

Function Decomposition

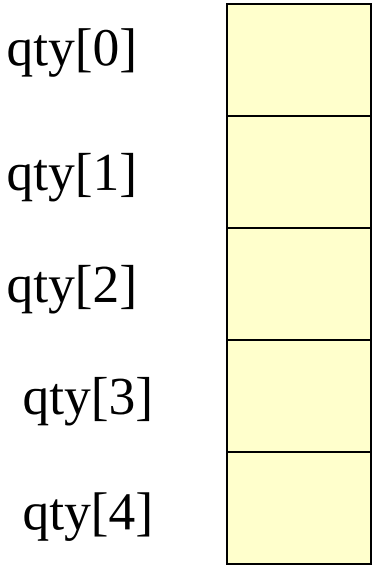
Passing primitives as arguments

Passing object references as arguments

Read
Pages 224 - 248

What will be the output? (Trace)

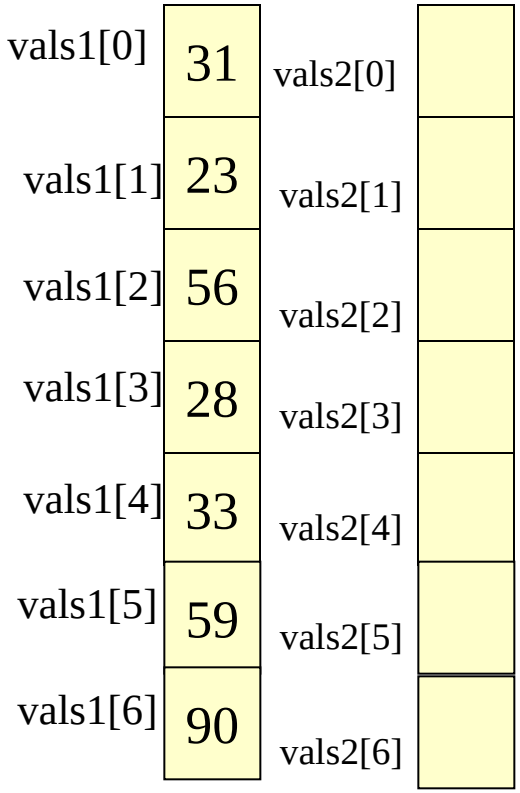
```
import java.util.*;
public class Array1
{  public static void main(String args[])
   {
      int qty[] = new int[5];

      for (int i=0; i<5; i++)
         qty[i] = i+1;
      for (int i=0; i<4; i++)
         qty[i+1] += qty[i];
      for (int i=0; i<5; i++)
         System.out.print(" " + qty[i]);
      System.out.println();
   }
}
```



Ans: _____

What will be the output? (Trace)

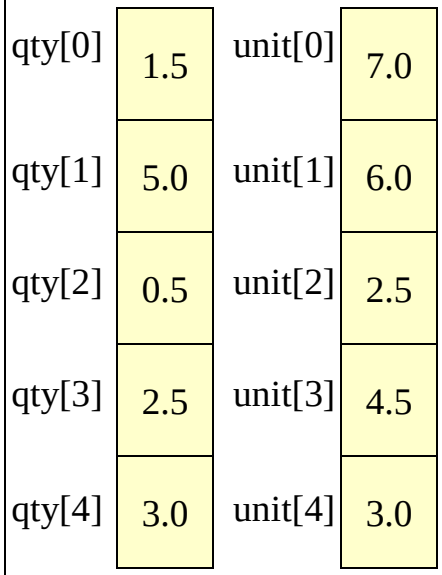
```
public class Array1B
{
   public static void main(String args[])
   {
      int vals1[] = {31,23,56,28,33,59,90};
       int vals2[] = new int[vals1.length];

       int j=0;
       for (int i=0; i<vals1.length; i++)
          if ( vals1[i] % 2 == 1)
              vals2[j++] = vals1[i];

       for (int k=0; k<j; k++)
          System.out.print(" " + vals2[k]);
   }
}
```



Arrays help us avoid declaring many variables

The quantity of 5 items purchased and their unit prices are stored in arrays qty and unit respectively. Complete the program below to find the total cost.

```
import java.util.*;
public class Array2
{  public static void main(String args[])
   {
      int vals[] = new int[5];
      double qty[] = {1.5,5.0,0.5,2.5,3.0};
      double unit[] = {7.0,6.0,2.5,4.5,3.0};

      double tCost=0;
      ...
      ...
      ...

      System.out.println("Total cost ="+tCost);

   }
}
```



Exercise (Manipulating Array)

- The Fibonacci series is given by

  1 1 2 3 5 8 …

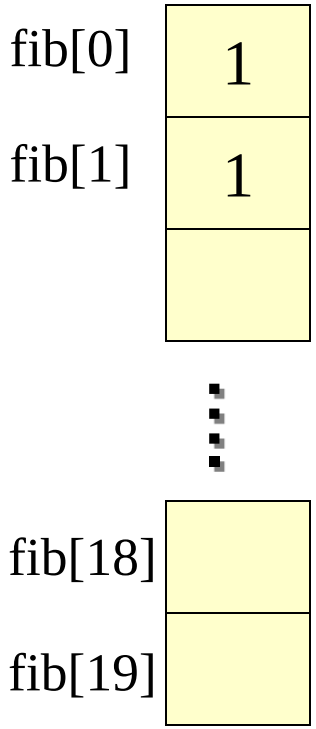
  (3rd and subsequent terms sum of previous terms)

- You are required to find the first 20 elements using an array which has first two elements set to 1.

- Finally print the array using a loop.

Exercise (Manipulating Array)

```
import java.util.*;
public class Array3
{
   public static void main(String args[])
   {
       int fib[] = new int[20];
      fib[0] = 1;
      fib[1] = 1;
       // Use loop to compute other 18 elements


       // Print all  elements using a loop

   }
}
```



Array of Strings

- You are required to write a program that allows users to enter up to 20 titles of books storing the values entered in array of String references.
- Then prompt user to enter the title he/she is searching. Display all matching titles (either partial or complete).
- For partial matching you may use indexOf(String s) method.

Program using Array of String

Manipulating Account objects without array

```
class Account {
  public Account(String accountID,
        String accountName, double amount)   {…}
  public void deposit(double amount) {…      }
  public boolean withdraw(double amount) {…}
  public double getBalance() {…}
  public boolean transfer(Account a, double amt){…}
  …
}
class TestAccount{
   Account mum = new Account("Mercy Brown",1000);
   Account dad = new Account("David Brown",2000);
   …
```

What if there are many such objects ? How can we keep track of their references ? Use an array or collection class.

Manipulating Account objects using array

```
import java.io.*;
public class TestAccounts {
  public static void main(String args[]) {
     BufferedReader stdin = new BufferedReader(
              new InputStreamReader(System.in));
     String name, accountID;
     double balance;
     Account[] accounts = new Account[3];
     for (int i=0;i<3; i++) {
       System.out.println("ID Account : ” +(i+1));
       accountID = stdin.readLine();
       System.out.println("Name Account : ”    +(i+1));
       name = stdin.readLine();
       System.out.println("Init.Bal. Account : ” +(i+1));
       balance = Double.parseDouble(stdin.readLine());
        accounts[i] = new Account(accountID,name,balance);
     }
```

```
        // debit monthly charges
        for (int i=0;i<3; i++) {
                accounts[i].withdraw(5.00);
        }

        // print all account balances
        for (int i=0;i<3; i++) {
           System.out.println("Balance for Account " + (i+1)
                        + " is " + accounts[i].getBalance());
        }
    }
}
```

Application : Menu Driven Program

Write a menu Driven Program to perform common operations with a double value as shown below. Initially the double value must be set to 10.

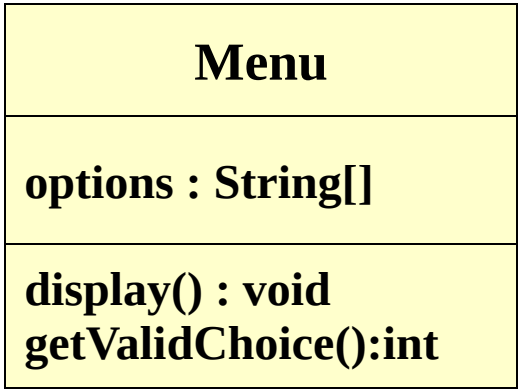
Menu

Read a new number	**1**
Print sqaure	**2**
Print cube	**3**
Print square root	**4**
Exit	**0**

Your Choice:

Menu class

- As part of this application you are required to create a class for Menu with specs given below. This class can be used in other exercises and assignments.
- Constructor which takes an array of String references, the menu options
- display method that displays menu strings and associated number (1,2…) corresponding to each menu option and 0 for exit.
- getValidChoice() method() which returns the valid integer selected. The valid integer must lie between 0 (for exit) and n, where n is the number of options.



Application: Manipulating Part objects using an array of references

- Construct an array of 5 Part (from last lecture) references
- Construct 5 Part objects specified below store their references in the array
- Allow user to repeatedly (using a loop) perform supply operations specifying quantity - print the total cost of purchase. Exit loop when user responds negatively to the prompt “Any more withdrawals? “.
- Finally, print all the Part details using a for loop.

ID	Name	Stock-level	Reorder-level	Unit-price
P123	Axle	78	30	120.00
P124	Shock absorber	32	15	180.00
P125	Brake Pad	325	120	72.50
P126	Exhaust pipe	170	80	155.50
P127	Front Panel	124	50	310.00

Manipulating Part objects (1)

Manipulating Part objects (2)

Manipulating Part objects (3)

Dealing with Dates

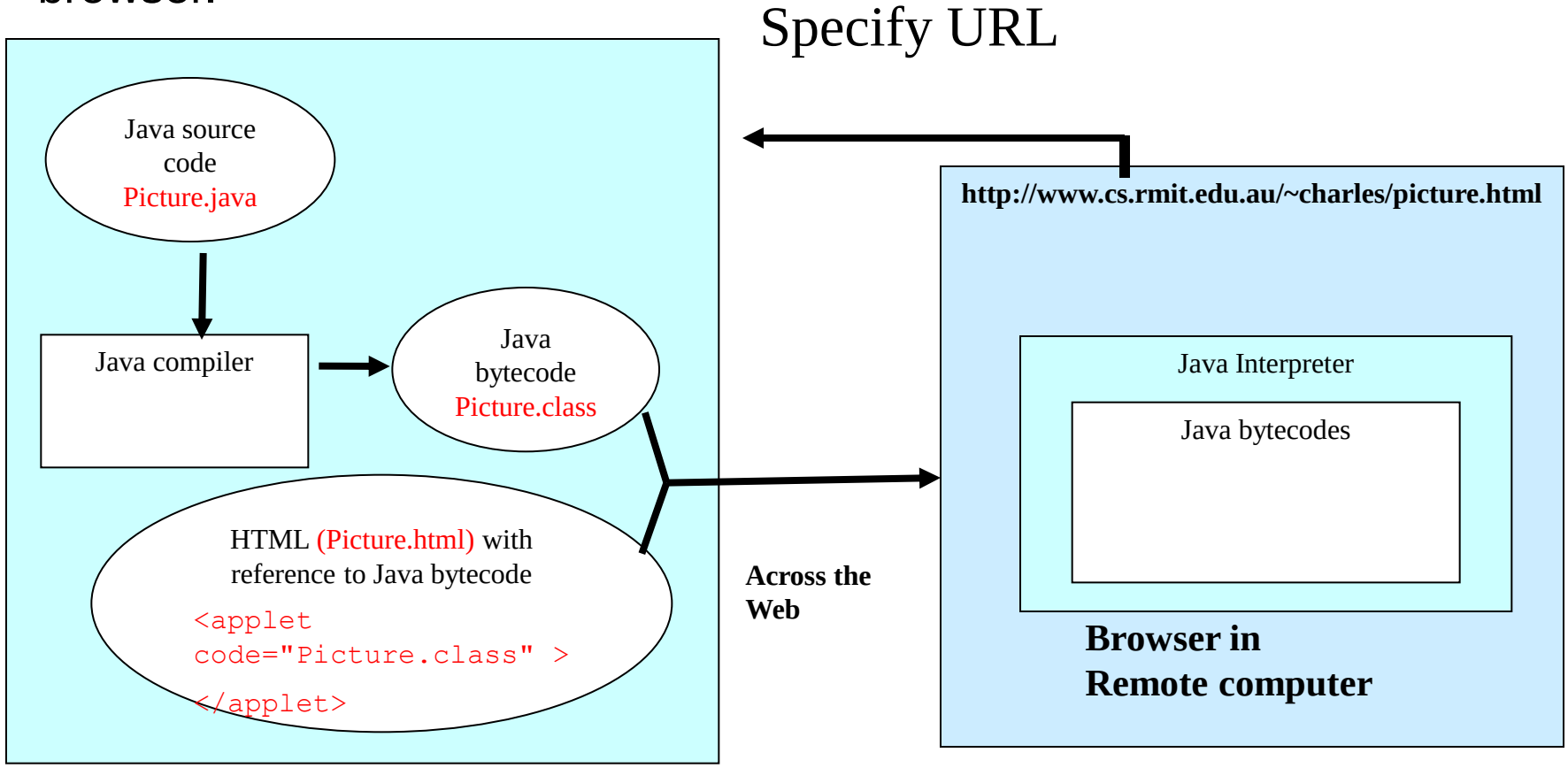
- The GregorianCalendar class can be used to set the date to specified date and time or current date and time.
- The method add(int field, int amount) to add or subtract specified amount to any calendar field (day,month, year, hour …)
- Method get(int field) returns specified calendar field.
- Method getTimeInMillis() return the time in ms since a set date (1st Jan 1970).
- Sample program sets date/time, advances date/time, finds the differences between date/time objects and prints specified calendar fields.

Dealing with Dates

```
import java.util.*;
public class TestDates
{  public static void main(String args[])
   {
      // GregorianCalendar date below correspond to 15th March 2007, 1.30 pm
      GregorianCalendar dateTime1 = new GregorianCalendar(2007,3,15,13,30);
      dateTime1.add(Calendar.MONTH,1); // advancing month to April
      dateTime1.add(Calendar.DAY_OF_MONTH,-5); // setting day to 10=15-5

     System.out.println("Year = " + dateTime1.get(Calendar.YEAR));
     System.out.println("Month = " + dateTime1.get(Calendar.MONTH));
     System.out.println("Day = " + dateTime1.get(Calendar.DAY_OF_MONTH));
     System.out.println("Hour = " + dateTime1.get(Calendar.HOUR_OF_DAY));
     System.out.println("Min = " + dateTime1.get(Calendar.MINUTE));
     System.out.println("Year = " + dateTime1.get(Calendar.YEAR));

     // Defaults constructor sets it to current  date and time
     GregorianCalendar dateTime2 = new GregorianCalendar();
     // Finding the difference in dates in millseconds
     long diff = dateTime1.getTimeInMillis() - dateTime2.getTimeInMillis();
     // converting difference to days
     int diffDays = (int)(diff / (1000 * 60 * 60 * 24));
     System.out.println("Difference in days = " + diffDays);
   }
}
```

Introduction to applets

- Two kinds of Java programs applets and applications.
- Applet (bytecode) is intended to be embedded into an HTML document, transported across the network and executed using a Web browser.

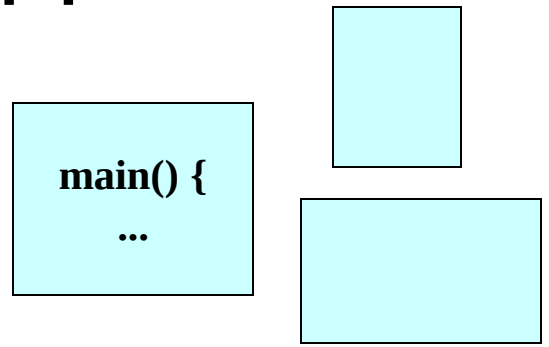


Introduction to applets

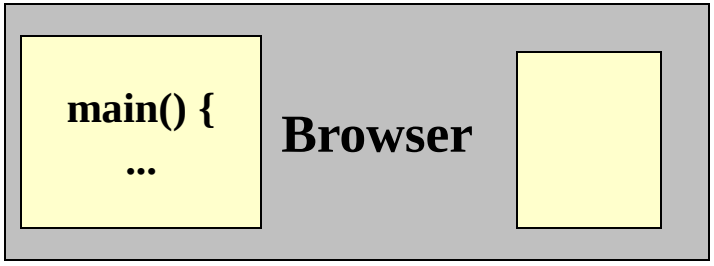
- So far, we have only looked at applications, the stand-alone programs that can be executed using the Java interpreter.
- Java applets can also be viewed locally using a Web browser or tool in Java's Software Development Kit, called appletviewer.

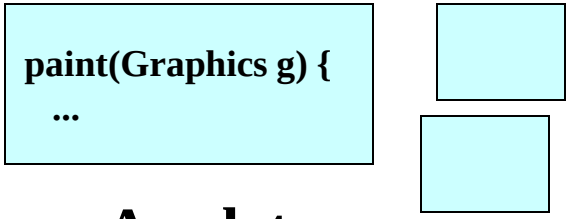
Differences Applets & Applications

- Unlike an application, applet is an incomplete program that is executed as part of another application, the browser.
- does not have a main() method, starting point for applications.
- The paint() method in the Applet derived class is automatically called whenever there is a need.



Application



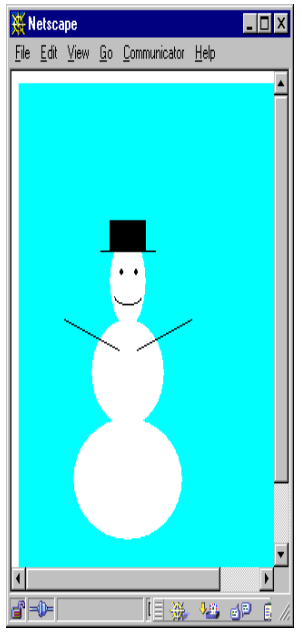


Applet

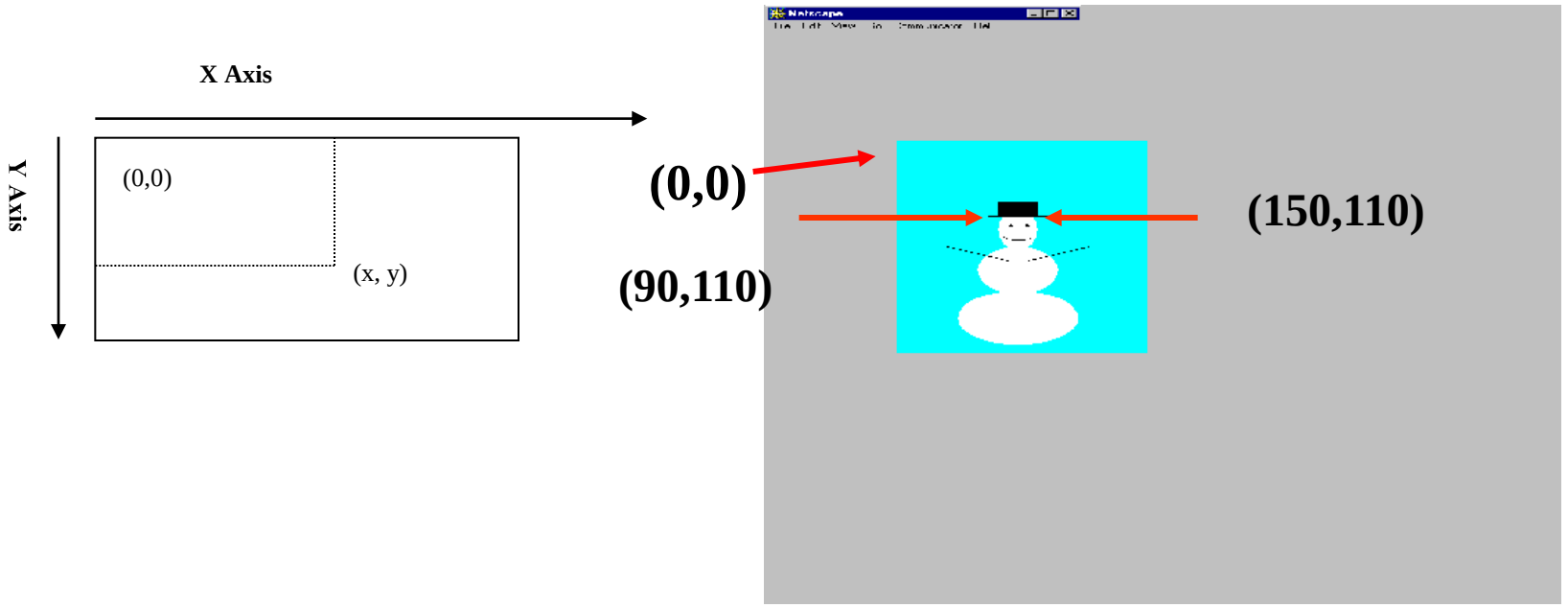
Creating a simple Applet

- **Derive a class from Applet**
- **Override the base class method paint(Graphics g)**
- **Draw on graphics object g (canvas) using its drawRect(),drawLine(), ...**

```
import java.applet.Applet;
import java.awt.*;

public class Picture  extends Applet {
    public void paint(Graphics g)   {
      …
      g.setColor(Color.black);
      g.drawLine(90,110,150,110);// hat
      g.fillRect(100,90,40,20);  // hat
      g.fillOval(111,122,4,4);   // eye
      …
   }
}
```



Java Coordinate System




```
g.drawLine(90,110,150,110);// hat
g.fillRect(100,90,40,20);  // hat
g.fillOval(111,122,4,4);   // eye
```

Commonly used methods of Graphics class

```
void setColor(Color  color)

drawString(String str, int x, int y)

void drawLine(int x1, int y1, int x2, int y2)

void drawOval(int x, int y, int width, int height)

void drawArc(int x, int y, int width, int height, int startAngle,
                                                    int endAngle)
```

Adding color to life

- To set color use void setColor(Color color)
- The predefined Color objects are objects are Color.black, Color.blue, Color.cyan, …
- Java uses the RGB color model, wherby you specify the color by the amounts (0.0F to 1.0F) of primary colors – red, green and blue.
- You may create your own color by mixing them in the required proportions as in

  Color magenta = new Color(1.0F, 0.0F, 1.0F);

Executing Applets using the Web

- **To transmit applet over the Web we need a (HTML) program.**
- **It contains tags specifying formatting instructions and identify special types such as applet.**
- **HTML file below uses an applet tag to indicates that the bytecode stored in Picture.class should be transported over the network and executed on the remote machine viewing it.**
- **Notice the tag also indicates the height and width of the applet.**
- **To view the applet you can use the command appletviewer Picture.html or use a browser.**

File Picture.html

```
<applet code="Picture.class" width=350, height=350>
</applet>
```

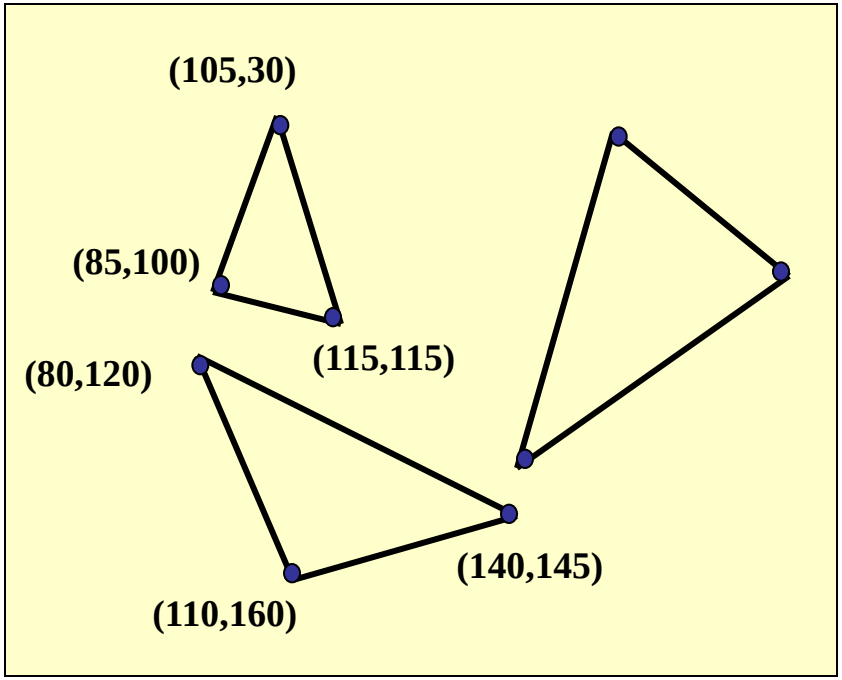
Method Decomposition

We are required to create a graphical application with 100's of triangles at user specified locations.

For each triangle we need to call drawLine() __ times.

Is there a better way ?

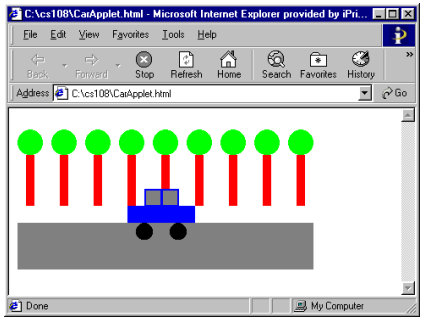
We can create a method say drawTriangle(), which takes the (x,y) coordinates of all 3 corners of that triangle. This method will then call drawLine() 3 times



Decomposing drawTriangle()

```
void drawTriangle(Graphics g, int x1,int y1,
                  int x2, int y2, int x3, int y3) {
      g.drawLine( ____________________________ );
      g.drawLine( ____________________________ );
      g.drawLine( ____________________________ );
}
//  paint method
drawTriangle(g,105,30,85,100,115,115);
drawTriangle(g,80,120,110,160,140,145);
```

An animation in 20 steps




```
//can write 20 methods - error prone - inefficient
drawCarPosition1(g); // draws at 1st position
drawCarPosition2(g); // draws at 2nd position
…
drawCarPosition20(g); // draws at 20th position
```

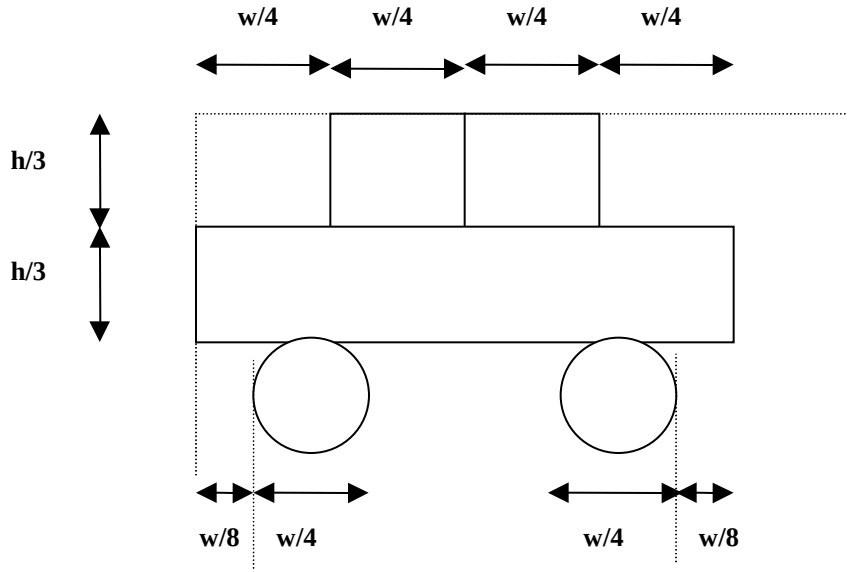
```
for (int i=0; i<20; i++){
    . . .
    drawCar(g, xpos, ypos, width, height, color);
}
```

Calls sub-methods

```
drawBody(g,...);
drawWindow(g,...);
drawWindow(g,...);
drawTyre(g,...);
drawTyre(g,...);
```

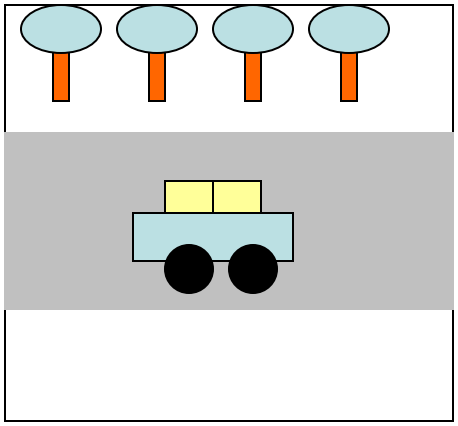
Decomposing drawCar()

h = height of car
w = width of car




```
public void drawCar(Graphics g, int x,int y, int width,
                                        int height, Color col)  {
        drawBody(g,x,y+height/3,width,height/3,col);
        drawWindow(g,x+width/4,y,width/4,height/3,col);
        drawWindow(g,x+width/2,y,width/4,height/3,col);
        drawTyre(g,x+width/8, y+2*height/3,width/4);
        drawTyre(g,x+5*width/8, y+2*height/3,width/4);
 }
```

Parameter Passing



paint(Graphicsg)

g

g,x,y,width,height,col

drawBackground(Graphics g)

drawCar(Graphics g, int x,int y,int width, int height, Color col)

g,x,y,width,height,col

drawTree(Graphics g, int x,int y,int width, int height)

g,x,y,width,height

g,x,y,width,height,col

drawBody(Graphics g, int x,int y,int width, int height, Color col)

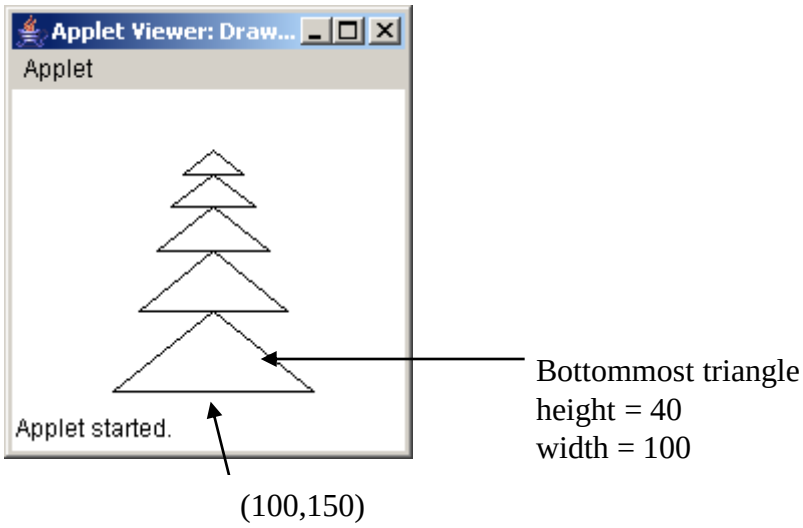
g,x,y,width,height,col

drawWindow(Graphics g, int x,int y,int width, int height, Color col)

drawTyre(Graphics g, int x,int y,int width, int height)

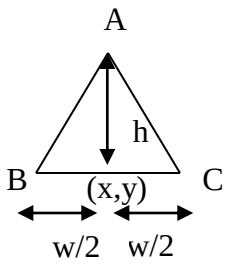
Applets and Method Passing: 2005 Exam Question

- You are required to create an applet with a series of 5 triangles as shown below. The centre of the base of the bottommost triangle is located at (100,150) with height 40 and width 100.
- For each subsequent triangle, the centre of the base of is located at the apex of the previous one, and the height and width are set to ¾ th of the previous one.
- All triangles have the same proportion as the isosceles triangle ABC (AB = AC) shown on the below left, with height h and width of base w.



- Write the method drawTriangle() below to draw a generic triangle with the values passed for the arguments x, y, w and h (see below).
- You may use the Graphics class method drawLine(int x1, int y1, int x2, int y2) to draw a line whose endpoints are (x1,y1) and (x2,y2).
- Then complete the paint() method below calling drawTriangle() repeatedly (using an appropriate loop) to draw the 5 triangles as shown above.

```
import java.awt.*;
import java.applet.*;
public class Triangles extends Applet
{  public void drawTriangle(Graphics g,int x, int y, int w, int h)
    {



    }

    public void paint(Graphics g)
    {



    }
}
```



Primitives & Object references as arguments

- When a method is called, the actual parameter values are computed and passed into the formal parameter variables.

- Thus when a primitive is passed, any changes made to the formal parameter (the copy) will not affect the actual parameter (original) in the calling method.

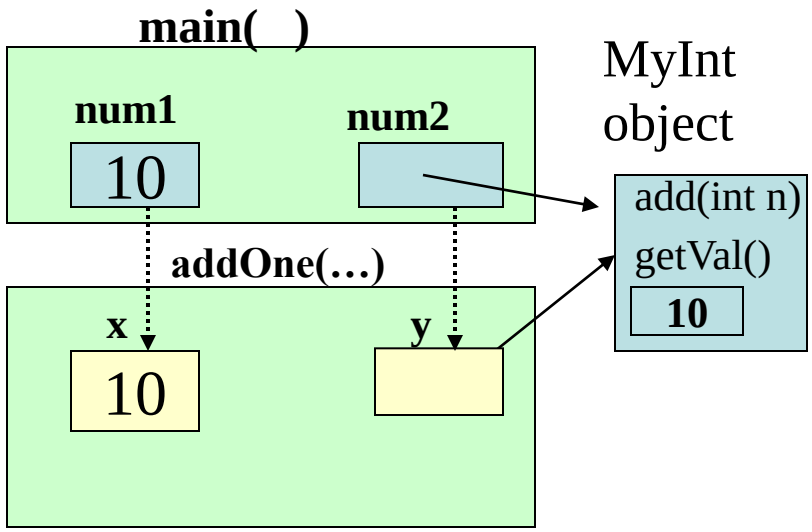
- However when a reference to an object is passed, the formal parameters in the called method can be used to directly manipulate the original object.

```
int num1 = 10;
MyInt num2 = new MyInt(10);
addOne(num1,num2);
…
```

Actual parameters

```
... addOne(int x, MyInt y)
{ . . .
```

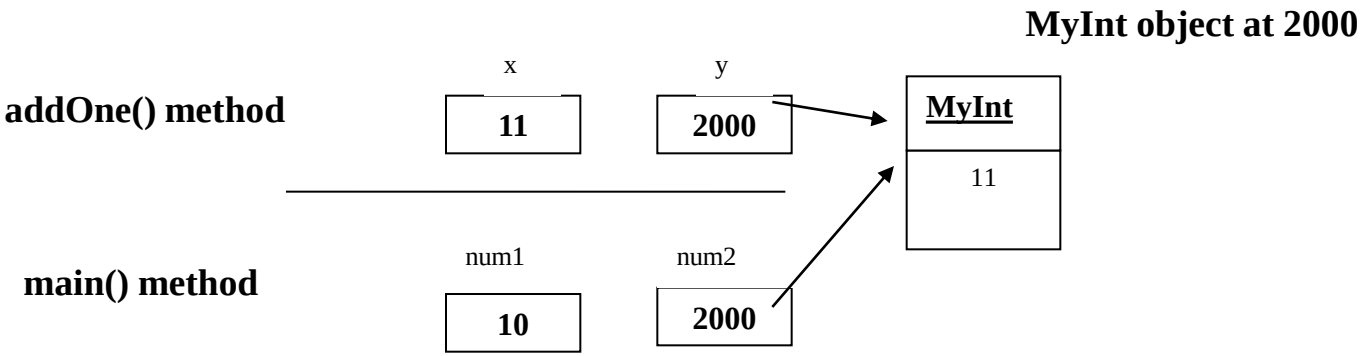
Formal parameters



Sample Program: Passing Primitives & Object references

```
class MyInt {
   public MyInt(int n) {      val = n;   }
   public void add(int n) {  val += n;    }
   public int getVal() {  return val;   }
   private int val;
}
public class PrimsAndObjects {
   public static void main (String[] args) {
      int num1 = 10;
      MyInt num2 = new MyInt(10);
      addOne(num1, num2);
      System.out.println("primitive num1 = " + num1+
        " object referred by num2 = " + num2.getVal() );
   }
   public static void addOne(int x, MyInt y) {
      x += 1;
      y.add(1);
   }
}
```

Memory snapshot in addOne() (after incrementing)



primitive num1 = 10 object referred by num2 = 11